EFFICIENT CONJUGATE GRADIENT BASED CHANNEL ESTIMATOR

Technical Field of the Invention

The present invention relates to a conjugate gradient based channel estimator.

Background of the Invention

Since the adoption of the ATSC digital television (DTV) standard in 1996 in the United States, there has been an ongoing effort to improve the design of receivers built for the ATSC DTV signal. The primary obstacle that faces designers in designing receivers so that they achieve good reception is the presence of multipath interference in the channel.

The broadcast television channel is a relatively severe multipath environment due to a variety of conditions that are encountered in the channel and at the receiver. Channels are characterized by impulse responses which may be several hundreds of symbols long, so that strong interfering signals may arrive at the receiver both before and after the largest amplitude signal. In addition, the signal transmitted through the channel is subject to time varying channel conditions due to the movement of the transmitter and signal reflectors, airplane flutter, and, for indoor reception, people

walking around the room. If mobile reception is desired, movement of the receiver must also be considered.

Moreover, the ATSC DTV signal uses trellis coded 8-level vestigial sideband (usually referred to as 8T-VSB or, more simply, as 8-VSB) as the modulation method. 8-VSB data symbols are real and have a signal pulse shape that is complex. Only the real part of the complex pulse shape is a Nyquist pulse. Therefore, even if there is no multipath, the imaginary part of the complex pulse shape contributes intersymbol interference (ISI) when the channel gain seen by the equalizer is not real.

Multipath and intersymbol interference adversely affects the ability of the receiver to correctly receive the symbols transmitted by the transmitter. Therefore, designers add equalizers to receivers in order to cancel the effects of multipath and intersymbol interference and thereby improve signal reception.

Because the channel is not known *a priori* at the receiver, the equalizer must be able to modify its response to match the channel conditions that it encounters and to adapt to changes in those channel conditions. To aid in the convergence of an adaptive

equalizer to channel conditions, the field sync segment of the frame as defined in the ATSC standard may be used as a training sequence for the equalizer. But when equalization is done in the time domain, long equalizers (those having many taps) are required due to the long channel impulse responses that characterize the channel.

The original Grand Alliance receiver used an adaptive decision feedback equalizer (DFE) with 256 taps. This adaptive decision feedback equalizer was adapted to the channel using a standard least mean square (LMS) algorithm, and was trained with the field sync segment of the transmitted frame. The LMS algorithm converges quite slowly and, even with only 256 taps, does not always converge during a single training sequence. Because the field sync segment is transmitted relatively infrequently (about every 260,000 symbols), the total convergence time of this equalizer is quite long if the equalizer only adapts on training symbols prior to convergence.

Therefore, in order to adapt equalizers to follow channel variations that occur between training sequences, the addition of blind and decision directed methods to equalizers has been suggested. However, when implemented in a realistic system, these methods may require several data fields to achieve convergence, and

convergence may not be achieved at all under difficult multipath conditions.

In any event, because multipath signals in the broadcast channel may arrive many symbols after the main signal, the decision feedback equalizer is invariably used in 8-VSB applications.

It has been argued that a blind decision feedback equalizer is required due to the rise in mean square error (MSE) between training sequences. However, adaptation of the trained equalizer in the simulation that supported this argument was frozen between training sequences. It is possible that a decision-directed equalizer with good tracking performance may be able to follow the channel variations tested.

Blind algorithms based on the Sato algorithm and on Godard's constant modulus algorithm (CMA) have been proposed. The error term in both of these algorithms uses a continuous blending of a decision-directed term with the blind term. This blending enables a smooth transition between the blind mode and the decision-directed mode. However, when implemented in a realistic system, these algorithms also may take several data fields to converge.

As mentioned previously, adaptive equalizers utilizing the least mean square (LMS) algorithm may converge slowly or not at all depending on the channel conditions. Convergence may be adversely affected if the input data auto-correlation matrix has a large eigenvalue spread. Also, if the decision feedback equalizer has not converged before the end of the training sequence, the shape of the objective function may change so that it includes local minima. These local minima may be caused by closed eye channel conditions and decision feedback equalizer error propagation.

The recursive least square (RLS) algorithm is well known to avoid these convergence problems. However, the recursive least square algorithm, in its basic form, requires the computationally intensive inversion of the input data auto-correlation matrix. Lattice based forms of the recursive least square algorithm avoid the need for this matrix inversion. However, the lattice based forms of the recursive least square algorithm are not easily amenable to the advantage of initialization from an initial channel impulse response (CIR) estimate, even though such an initialization may be desirable.

Recent work in reduced rank filtering has connected the multi-stage nested Wiener filter (MSNWF) of

Goldstein and Reed to the conjugate gradient algorithm (CG). It has been shown that the multi-stage nested Wiener filter solves the Wiener-Hopf equations in the Krylov subspace associated with the auto-correlation matrix and the cross-correlation vector. The multi-stage nested Wiener filter is then re-formulated using the Lanczos iteration. It has also been shown that the Lanczos-based multi-stage nested Wiener filter is equivalent to the conjugate gradient algorithm. Since the multi-stage nested Wiener filter often needs few dimensions to approach the performance of the full-rank Wiener filter, the conjugate gradient algorithm is a good candidate for an adaptive equalization algorithm with fast convergence.

A description of the conjugate gradient optimization algorithm and many of its mathematical properties may be found in "An Introduction to Optimization," by E. K. P. Chong and S. Zak, New York, NY, John Wiley & Sons, 1996. The algorithm described is applicable to the optimization of a fixed objective function. An excellent review of adaptive filtering with the conjugate gradient algorithm is found in "Analysis of conjugate gradient algorithms for adaptive filtering," by P. S. Chang and A. N. Willson, Jr., vol. 48, pp. 409-418,

IEEE Transactions on Signal Processing, February 2000. The mathematical richness of the algorithm relationships leads to a variety of options in the implementation of the basic conjugate gradient algorithm for minimizing a quadratic objective function. These options can be carried over to the adaptive situation to provide a number of options for implementation of the algorithm. One characteristic of the algorithm which may also be exploited is that it works directly with the correlation matrices and vectors. This characteristic makes initialization based on an initial channel impulse response straightforward, assuming information is available for this purpose.

Like lattice based forms of the recursive least square algorithm, the conjugate gradient algorithm does not require the inversion of the input data auto-correlation matrix. Another characteristic of the conjugate gradient algorithm may be exploited by recognizing that the input data auto-correlation matrix is the product of two Toeplitz matrices.

The invention described in U.S. Patent Application Serial No. (7227) uses these characteristics to substantially reduce the computation load in the conjugate gradient algorithm and to even eliminate the

need to form the input auto-correlation matrix. Accordingly, the improved conjugate gradient algorithm disclosed in that application may be used to achieve satisfactory convergence times for equalizers.

The (7227) application also suggests that the conjugate gradient algorithm can be used to estimate channels. The present invention reduces the computation complexity in performing the conjugate gradient algorithm with respect to channel estimators.

Summary of the Invention

In accordance with one aspect of the present invention, a method of processing a received signal y to produce a channel estimate comprises the following: (a) decoding the received signal y to form data s; (b) forming a convolution matrix $\hat{S}$ from the data s; (c) forming a matrix F from the data s, wherein the matrix F results from forming the matrix $\hat{S}$ as a convolution matrix; and, (d) performing a conjugate gradient algorithm to determine the channel estimate, wherein the conjugate gradient algorithm is based on the received signal y, the matrix $\hat{S}$, and the matrix F.

In accordance with another aspect of the present invention, a method of processing a received

signal y comprises the following: (a) decoding the received signal y to form data s; (b) forming a convolution matrix $\hat{S}$ from the data s; (c) forming a matrix F from the data s, wherein the matrix F results from forming the matrix $\hat{S}$ as a convolution matrix; and, (d) performing a conjugate gradient algorithm based on the received signal y, the matrix $\hat{S}$, and the matrix F.

In accordance with still another aspect of the present invention, a method of processing a received signal y comprises the following: (a) decoding the received signal y to form data s; (b) forming a convolution matrix $\hat{S}$ from the data s; (c) forming a matrix F from the data s, wherein the matrix F results from forming the matrix $\hat{S}$ as a convolution matrix; and, (d) performing a conjugate gradient algorithm based on the received signal y, the matrix $\hat{S}$, and the matrix F, wherein the conjugate gradient algorithm includes forming FFTs based on the received signal y, the matrix $\hat{S}$, and the matrix F, multiplying the FFTs to form a multiplication product, and forming an inverse FFT of the multiplication product.

In accordance with yet another aspect of the present invention, a method of processing a received signal y to produce a channel estimate comprises the

following: (a) decoding the received signal y to form data s; (b) forming a convolution matrix $\hat{S}$ from the data s; (c) forming a matrix F from the data s, wherein the matrix F results from forming the matrix $\hat{S}$ as a convolution matrix; and, (d) performing a conjugate gradient algorithm to determine the channel estimate, wherein the conjugate gradient algorithm is based on the received signal y, the matrix $\hat{S}$, and the matrix F, and wherein the conjugate gradient algorithm includes forming FFTs based on the received signal y, the matrix $\hat{S}$, and the matrix F, multiplying the FFTs to form a multiplication product, and forming an inverse FFT of the multiplication product.

Brief Description of the Drawings

These and other features and advantages will become more apparent from a detailed consideration of the invention when taken in conjunction with the drawings in which:

Figure 1 illustrates a filter that can be used in a decision feedback equalizer according to an embodiment of the present invention;

Figure 2 illustrates the decision feedback equalizer according to an embodiment of the present invention;

Figure 3 illustrates a matrix S which is derived from a vector of symbol decisions of length 2496 and is useful in describing the present invention;

Figure 4 illustrates the decomposition of the matrix S to form two matrices $\hat{\mathbf{S}}$ and F useful in implementing the present invention; and,

Figure 5 shows a system for determining a channel estimate h based on the matrices $\hat{\mathbf{S}}$ and F.

Detailed Description

As shown in Figure 1, a filter 10 can be used as an equalizer and includes storage registers 12[n], ..., 12[n+N-3], 12[n+N-2], 12[n+N-1]. For example, N may be 512. The storage registers 12[n], ..., 12[n+N-3], 12[n+N-2], 12[n+N-1] store corresponding received data y[n], ..., y[n+N-3], y[n+N-2], y[n+N-1]. The output of each of the storage registers 12[n], ..., 12[n+N-3], 12[n+N-2], 12[n+N-1] is coupled to a corresponding multiplier 14[n], ..., 14[n+N-3], 14[n+N-2], 14[n+N-1]. Accordingly, the multipliers 14[n], ..., 14[n+N-3], 14[n+N-2], 14[n+N-1] receive the stored data from the

storage registers 12[n], ..., 12[n+N-3], 12[n+N-2], 12[n+N-1]. The multipliers 14[n], ..., 14[n+N-3], 14[n+N-2], 14[n+N-1] also receive corresponding tap weights g[n], ..., g[n+N-3], g[n+N-2], g[n+N-1]. The multipliers 14[n], ..., 14[n+N-3], 14[n+N-2], 14[n+N-1] multiply each of the stored data y[n], ..., y[n+N-3], y[n+N-2], y[n+N-1] by a corresponding one of the tap weights g[n], ..., g[n+N-3], g[n+N-2], g[n+N-1], and the multiplication results are summed by a summer 16 to provide the output of the filter 10. Various algorithms as described above have been implemented in order to set the values of the tap weights g[n], ..., g[n+N-3], g[n+N-2], g[n+N-1].

A decision feedback equalizer 20 is shown in Figure 2 and includes a feed forward filter 22, a feedback filter 24, a summer 26, a hard decision device 28, and a controller 30 that computes the tap weights for the feed forward filter 22 and the feedback filter 24. Each of the feed forward filter 22 and the feedback filter 24 may comprise the filter 10 of Figure 1. The hard decision device 28, for example, may be a decoder or a slicer.

The output $\hat{s}[n]$ of the summer 26 is the equalized output of the decision feedback equalizer 20.

The hard decision device 28 (such as a slicer) determines the closest value $\tilde{s}[n]$ for each of the symbols provided as the output $\hat{s}[n]$ of the summer 26. These values $\tilde{s}[n]$ are fed back as inputs to the feedback filter 24. The feedback filter 24 applies tap weights $\hat{g}_B[n]$ to the values $\tilde{s}[n]$ and provides its output to the summer 26. Likewise, the feed forward filter 22 applies tap weights $\hat{g}_F[n]$ to the values y[n] of the received signal and provides its output to the summer 26. The summer 26 sums the output of the feedback filter 24 with the output of the feed forward filter 22 in order to produce the output $\hat{s}[n]$.

The output $\hat{s}[n]$ of the decision feedback equalizer 20 is given by the following equation:

$$\begin{aligned}\hat{s}[n] &= \mathrm{Re}\left\{g_F^h y[n]\right\} + g_B^T \tilde{s}[n] \\ &= \mathbf{g}_{FR}^T \mathbf{y}_R[n] + \mathbf{g}_{FI}^T \mathbf{y}_I[n] + \mathbf{g}_B^T \tilde{\mathbf{s}}[n] \end{aligned} \quad (1)$$

where $\tilde{\mathbf{s}}[n]$ as indicated above is defined to be the symbol estimates provided by the hard decision device 28, where $\mathbf{g}_{FR} = \mathrm{Re}\{\mathbf{g}_F\}$ is the real part of the tap weights $\mathbf{g}_F$ of the feed forward filter 22, where $\mathbf{g}_{FI} = \mathrm{Im}\{\mathbf{g}_F\}$ is the imaginary part of the tap weights $\mathbf{g}_F$ of the feed forward filter 22,

and where $\mathbf{g}_B$ are the tap weights of the feedback filter 24.

The definitions given by the following equations may be assumed:

$$\tilde{\mathbf{y}}_F[n] = \left[\mathbf{y}_R^T[n]\mathbf{y}_I^T[n]\right]^T \quad (2)$$

$$\tilde{y}[n] = \left[\tilde{y}_F^T[n]\tilde{s}^T[n]\right]^T \quad (3)$$

$$\tilde{g}_F[n] = \left[g_{FR}^T[n]g_{FI}^T[n]\right]^T \quad (4)$$

$$\tilde{g}[n] = \left[\tilde{g}_F^T[n]\tilde{g}_B^T[n]\right]^T \quad (5)$$

where $\mathbf{y}_R = \mathrm{Re}\{\mathbf{y}_F\}$ is the real part of the received symbols $\mathbf{y}_F$, and where $\mathbf{y}_I = \mathrm{Im}\{\mathbf{y}_F\}$ is the imaginary part of the received symbols $\mathbf{y}_F$. Then, the equalized symbols $\hat{\mathbf{s}}[n]$ provided by the decision feedback equalizer 20 are given by the following equation:

$$\begin{aligned}\hat{\mathbf{s}}[n] &= \tilde{\mathbf{g}}_F^T\tilde{\mathbf{y}}_F[n] + \mathbf{g}_B^T\tilde{\mathbf{s}}[n] \\ &= \tilde{\mathbf{g}}^T\tilde{\mathbf{y}}[n]\end{aligned} \quad (6)$$

The equalizer input $\mathbf{y}[n]$ and the feed forward tap weights $\mathbf{g}_F$ are complex values, the feedback tap weights $\mathbf{g}_B$ are real, and the output $\hat{\mathbf{s}}[n]$ of the decision feedback equalizer 20 is real. The imaginary part of the output of the feed forward filter 22, $\mathbf{g}_{FI}$, may be ignored and, therefore, not computed.

It is desired to determine the tap weight vector $\tilde{\mathbf{g}}[n]$ so that the error at the output of the decision feedback equalizer 20 is minimized according to the following equations:

$$\tilde{\mathbf{g}}[n] = \arg\min_{\mathbf{g}} \frac{1}{2} E\left\{(s[n] - \hat{s}[n])^2\right\} \tag{7}$$

$$F(g) = E\left\{(s[n] - \hat{s}[n])^2\right\}$$

$$= \sigma_s^2 - \tilde{\mathbf{g}}^H[n]\mathbf{r}_{s\tilde{y}} - \mathbf{r}_{s\tilde{y}}^H\tilde{\mathbf{g}}[n] + \tilde{\mathbf{g}}^H[n]R_{\tilde{y}\tilde{y}}\mathbf{g}[n] \tag{8}$$

where σ_s^2 is the symbol variance and is equal to the average symbol energy, $R_{\tilde{y}\tilde{y}}$ is an autocorrelation matrix, and $\mathbf{r}_{s\tilde{y}}$ is a cross correlation vector. The autocorrelation matrix $R_{\tilde{y}\tilde{y}}$ and the cross correlation vector $\mathbf{r}_{s\tilde{y}}$ are defined in detail hereinafter. The error function F(g) is minimized by setting its gradient to zero according to the following equation:

$$\nabla F(g) = R_{\tilde{y}\tilde{y}}\tilde{\mathbf{g}}[n] - \mathbf{r}_{s\tilde{y}} = 0 \qquad (9)$$

This minimization leads to the following well known Weiner-Hopf equation:

$$R_{\tilde{y}\tilde{y}}\tilde{\mathbf{g}}[n] = \mathbf{r}_{s\tilde{y}} \qquad (10)$$

Solving for the desired tap weights $\tilde{\mathbf{g}}[n]$ leads to the following equation:

$$\tilde{\mathbf{g}}[n] = R_{\tilde{y}\tilde{y}}^{-1}\mathbf{r}_{s\tilde{y}} \qquad (11)$$

So, in order to calculate the desired tap weight vector, the large system of equations represented by equation (11) must be solved. This solution requires the inversion of the large matrix $R_{\tilde{y}\tilde{y}}$ and, therefore, a considerable amount of computation.

One method for reducing the amount of computation involved in solving a large system of equations is the aforementioned conjugate gradient algorithm. In order to set up equation (11), estimation of the auto-correlation matrix $R_{yy}[n=0]$ and the cross correlation vector $\mathbf{r}_{sy}[n=0]$ is required. By definition,

the auto-correlation matrix $R_{yy}[n]$ is given by the following equation:

$$R_{\tilde{y}\tilde{y}}[n] = E\left[\tilde{\mathbf{y}}[n]\tilde{\mathbf{y}}^{H}[n]\right] \tag{12}$$

and the cross correlation vector $\mathbf{r}_{sy}[n]$ is given by the following equation:

$$\mathbf{r}_{s\tilde{y}}[n] = E\left[s[n]\tilde{\mathbf{y}}[n]\right] \tag{13}$$

where s[n] represents the transmitted symbols, E is the expectation operator, and H denotes the Hermitian transpose.

The auto-correlation matrix $R_{\tilde{y}\tilde{y}}[n]$ and the cross correlation vector $\mathbf{r}_{s\tilde{y}}[n]$ can be estimated using the forgetting factor method according to the following equations:

$$\begin{aligned}\hat{\mathbf{R}}_{\tilde{y}\tilde{y}}[n] &= \sum_{m=0}^{n-1} \gamma^{m}\tilde{\mathbf{y}}[n-m]\,\tilde{\mathbf{y}}^{H}[n-m] \\ &= \gamma\hat{\mathbf{R}}_{yy}[n-1] + \tilde{\mathbf{y}}[n]\tilde{\mathbf{y}}^{H}[n] \qquad (14) \\ \hat{\mathbf{r}}_{s\tilde{y}}[n] &= \sum_{m=0}^{n-1} \gamma^{m}\mathbf{s}[n-m]\,\tilde{\mathbf{y}}^{H}[n-m]\end{aligned}$$

$$= \gamma \hat{\mathbf{r}}_{s\tilde{y}}[n-1] + \mathbf{s}[n]\tilde{\mathbf{y}}^{H}[n] \qquad (15)$$

where $0 < \gamma < 1$ is the forgetting factor. If γ is set to 0, the equations (14) and (15) yield the estimates of the auto-correlation matrix $\hat{R}_{\tilde{y}\tilde{y}}$ and the cross correlation vector $\hat{\mathbf{r}}_{s\tilde{y}}$ from the most recent sample vector $\mathbf{y}$.

In determining the tap weights $\mathbf{g}_F$ for the feed forward filter 22 and $\mathbf{g}_B$ for the feedback filter 24, the controller 30 initializes the auto-correlation matrix $R_{\tilde{y}\tilde{y}}[0]$ using equation (14) and the cross correlation vector $\mathbf{r}_{s\tilde{y}}[0]$ using equation (15) by performing averaging over a period of time such as 1/2 data field or so. The controller 30 also initializes the tap weights $\tilde{g}_0[0] = 0$ and initializes the decision vector $\mathbf{s}[0]$ to the first N_{fb} of the known training symbols that are contained in the field sync portion of each data field as discussed above.

Further, the controller 30 initializes the data vector $\tilde{\mathbf{y}}[0]$ as follows. The data vector $\tilde{\mathbf{y}}_F[0]$ as given in equation (2) is initialized by setting the real part $\mathbf{y}_R[n = 0] = \{y_R[0], \ldots, y_R[N_{ff} - 1]\}^T$ such that $y_R[0], \ldots, y_R[N_{ts} - N_{fb} - 1]$ are the real part of the remaining training data (after those in $\tilde{s}[0]$) and

$y_R[N_{ts} - N_{fb}], \Lambda, y_R[N_{ff} - 1]$ are the real part of the received data after the training sequence. This setting is repeated for $\tilde{\mathbf{y}}_I[0]$. Then, $\tilde{\mathbf{y}}_F[0] = \left[\mathbf{y}_R^T[0]\mathbf{y}_I^T[0]\right]^T$ is constructed as given in equation (2) and $\tilde{\mathbf{y}}[0] = \left[\tilde{\mathbf{y}}_F^T[0]\tilde{\mathbf{s}}^T[0]\right]^T$ is constructed as given in equation (3) based on the initialized vectors $\tilde{\mathbf{y}}_F[0]$, $\tilde{\mathbf{y}}_R[0]$, and $\tilde{\mathbf{y}}_I[0]$ as given above. In an ATSC digital television environment, for example, the number of training symbols N_{ts} may be 728, the number of feed forward taps N_{ff} may be 512, and the number of feedback taps N_{fb} may also be 512.

Further, the controller 30 initializes the output $\hat{s}[n]$ of the decision feedback equalizer 20 as $\hat{s}[0] = \tilde{\mathbf{g}}_0^T[0]\tilde{\mathbf{y}}[0]$ in accordance with equation (6), the controller 30 initializes a gradient $\mathbf{t}_n[n]$ as $\mathbf{t}_0[0] = \hat{R}_{\tilde{y}\tilde{y}}[0]\tilde{\mathbf{g}}_0[0] - \hat{\mathbf{r}}_{sy}[0]$, and the controller 30 sets the variable n to 0.

The controller 30 then executes a conjugate gradient algorithm once for each received data element, where each iteration of the algorithm is initiated upon the shifting of a new received data element into the feed forward filter 22. The parameters of the conjugate gradient algorithm as described above are defined in the table.

Table

Parameter Name	Equalizer Notation
Object Function Argument (Equalizer)	$\mathbf{g}_k[n]$
Argument Step Size	$\alpha_k[n]$
Gradient	$\mathbf{t}_k[n]$
Conjugate Directions	$\mathbf{b}_k[n]$
Conjugate Direction Step Size	$\beta_k[n]$

This conjugate gradient algorithm is as follows:

1. calculate conjugate direction $b_k[n]$

   if n modulo(N_{taps})= 0

   $$\mathbf{b}_0[n] = -\mathbf{t}_0[0] \qquad \text{(reset)}$$

   else

   $$\beta_0[n] = \frac{\left(\mathbf{t}_o[n] - \mathbf{t}_{-1}[n]\right)^H \mathbf{t}_0[n]}{\left(\mathbf{t}_0[n] - \mathbf{t}_{-1}[n]\right)^H \mathbf{b}_{-1}[n]}$$

   $$\mathbf{b}_0[n] = -\mathbf{t}_0[n] + \beta_0[n]\mathbf{b}_{-1}[n]$$

2. calculate argument step size $\alpha_k[n]$

   $$\alpha_0[n] = \frac{-\mathbf{b}_0^H[n]\mathbf{t}_0[n]}{\mathbf{b}_0^H[n]\hat{\mathbf{R}}_{\tilde{y}\tilde{y}}[n]\mathbf{b}_0[n]}$$

3. tap update

$$\tilde{\mathbf{g}}_1[n] = \tilde{\mathbf{g}}_0[n] + \alpha_0[n]\mathbf{b}_0[n]$$

4. $n = n + 1$

$$\tilde{\mathbf{g}}_0[n] = \tilde{\mathbf{g}}_1[n-1]$$

$$\mathbf{t}_{-1}[n] = \mathbf{t}_0[n-1]$$

$$\mathbf{b}_{-1}[n] = \mathbf{b}_0[n-1]$$

5. update input vector $\tilde{\mathbf{y}}[n]$ with the next received data element and update $\hat{R}_{\tilde{y}\tilde{y}}[n]$ per equation (14)

6. Calculate output of the decision feedback equalizer 20

$$\hat{s}[n] = \tilde{g}_0^T[n]\tilde{\mathbf{y}}[n]$$

7. calculate gradient $t_k[n]$

$$\mathbf{t}_0[n] = \mathbf{t}_{-1}[n] + \alpha_0[n]\left[\hat{R}_{\tilde{y}\tilde{y}}[n]\mathbf{b}_0[n]\right]$$

8. Return to step 1

The largest of the part of the computations required by this conjugate gradient algorithm is contributed by the matrix-vector multiplications $\hat{R}_{\tilde{y}\tilde{y}}[n]\mathbf{b}_0[n]$ of steps 2 and 7. These multiplications require the update of matrix $\hat{R}_{\tilde{y}\tilde{y}}[n]$ from the vector $\tilde{\mathbf{y}}[n]$ as in equation (14). This update requires one multiply for each of the N_{taps}^2 matrix elements, a total of N_{taps}^2 multiplies. Then, the matrix-vector multiplication

requires an additional N_{taps}^2 multiplies. Accordingly, the total computation is $N_{taps}^2 + N_{taps}^2 = 2N_{taps}^2$ multiplies.

This computational load can be decreased as explained below.

A matrix-vector multiplication given by **a** = **Yb**, where **Y** is a convolution matrix, **b** is a vector, and **a** is the multiplication result, is equivalent to a[n] = y[n] * b[n] where * denotes linear convolution. These equations may be represented in matrix form by the following equation:

$$\begin{bmatrix} a[0] \\ a[1] \\ a[2] \end{bmatrix} = \begin{bmatrix} y[0] & y[-1] & y[-2] \\ y[1] & y[0] & y[-1] \\ y[2] & y[1] & y[0] \end{bmatrix} \begin{bmatrix} b[0] \\ b[1] \\ b[2] \end{bmatrix} \qquad (16)$$

where n is chosen as a small number for explanatory purposes only. The columns and rows of the matrix **Y** in equation (16) can be circularly extended in accordance with the following equation:

$$\begin{bmatrix} a[0] \\ a[1] \\ a[2] \\ dc \\ dc \end{bmatrix} = \begin{bmatrix} y[0] & y[-1] & y[-2] & y[2] & y[1] \\ y[1] & y[0] & y[-1] & y[-2] & y[2] \\ y[2] & y[1] & y[0] & y[-1] & y[-2] \\ y[-2] & y[2] & y[1] & y[0] & y[-1] \\ y[-1] & y[-2] & y[2] & y[1] & y[0] \end{bmatrix} \begin{bmatrix} b[0] \\ b[1] \\ b[2] \\ 0 \\ 0 \end{bmatrix} \qquad (17)$$

where dc means don't care. Equation (17) is equivalent to a[n] = y[n] ⊗ b[n] where ⊗ denotes circular convolution. Circular convolution is equivalent to the following operation:

$$\mathcal{Y} = \mathrm{FFT}(\mathbf{y}) \tag{18}$$

$$\mathcal{B} = \mathrm{FFT}(\mathbf{b}) \tag{19}$$

$$\mathcal{A} = \mathcal{Y} \times \mathcal{B} \tag{20}$$

$$\mathbf{a} = \mathrm{IFFT}(\mathcal{A}) \tag{21}$$

where FFT in equations (18) and (19) indicates the Fast Fourier Transform operation, IFFT in equation (21) indicates the Inverse Fast Fourier Transform operation, and x in equation (20) indicates point-wise vector multiplication such that the elements of the resultant vector are given by $\mathbf{Y}_0 \times \mathbf{B}_0$, $\mathbf{Y}_1 \times \mathbf{B}_1$, $\mathbf{Y}_2 \times \mathbf{B}_2$, etc.

An embellishment on the above process involves 0 padding the columns to lengths of 2^n. This 0 padding, as is well known, permits a computationally efficient FFT. Accordingly, the matrix-vector multiplication **a** = **Yb** can be written with 0 padding in matrix form according to the following equation:

$$\begin{bmatrix} a[0] \\ a[1] \\ a[2] \\ dc \\ dc \\ dc \\ dc \\ dc \end{bmatrix} = \begin{bmatrix} y[0] & y[-1] & y[-2] & 0 & 0 & 0 & y[2] & y[1] \\ y[1] & y[0] & y[-1] & y[-2] & 0 & 0 & 0 & y[2] \\ y[2] & y[1] & y[0] & y[-1] & y[-2] & 0 & 0 & 0 \\ 0 & y[2] & y[1] & y[0] & y[-1] & y[-2] & 0 & 0 \\ 0 & 0 & y[2] & y[1] & y[0] & y[-1] & y[-2] & 0 \\ 0 & 0 & 0 & y[2] & y[1] & y[0] & y[-1] & y[-2] \\ y[-2] & 0 & 0 & 0 & y[2] & y[1] & y[0] & y[-1] \\ y[-1] & y[-2] & 0 & 0 & 0 & y[2] & y[1] & y[0] \end{bmatrix} \begin{bmatrix} b[0] \\ b[1] \\ b[2] \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad (22)$$

It is noted that matrix **Y** has a Toeplitz structure, and that Y^H is also a Toeplitz matrix. (H denotes Hermitian transpose.)

It will be shown that the need for both the $R_{\tilde{y}\tilde{y}}$ update and the $R_{\tilde{y}\tilde{y}}\mathbf{b}$ multiply can be eliminated, and that the matrix $R_{\tilde{y}\tilde{y}}$ need not even be created.

Let $\gamma = 0$ for equation (14). It is well known that

$$R_{\tilde{y}\tilde{y}} = \tilde{\mathbf{y}}\tilde{\mathbf{y}}^H = YY^H$$

where Y is a convolution matrix based on the vector **y** (that is, **y** forms the first column of the matrix Y). The vector $\tilde{\mathbf{y}}$ may be 0 padded to length 2^n. Let $\tilde{\mathbf{y}}'$ be the first column of Y^H. For the conjugate gradient operation, the calculation given by the following equation must be made:

$$\boldsymbol{\nu} = \hat{R}_{\tilde{y}\tilde{y}}[n]\mathbf{b}_0[n] = Y^H Y\mathbf{b}_0[n] \quad (23)$$

In order to calculate $\boldsymbol{\nu}$, the quantity $Y'\mathbf{b}_0[n]$ should first be calculated according to the following equations:

$$\mathcal{Y} = \mathrm{FFT}(\mathbf{y}) \quad (24)$$

$$\mathcal{B} = \mathrm{FFT}(\mathbf{b}) \quad (25)$$

$$\mathcal{X} = \mathcal{Y} \times \mathcal{B} \quad (26)$$

$$\mathbf{x} = \mathrm{IDFT}(\mathcal{X}) \quad (27)$$

$$\mathbf{x}_N = \text{first N elements of } \mathbf{x}, \text{ remaining elements} = 0 \quad (28)$$

$$\mathcal{X}_N = \mathrm{FFT}(\mathbf{x}_N) \quad (29)$$

where x in equation (26) denotes point-wise vector multiplication. Then, $Y \times \mathcal{X}_N$ is calculated according to the following equations:

$$\mathcal{Y}' = \mathrm{FFT}(\mathbf{y}') \quad (30)$$

$$\mathcal{V} = \mathcal{Y}' \times \mathcal{X} \quad (31)$$

$$\boldsymbol{\nu} = \mathrm{IDFT}\mathcal{V} \quad (32)$$

$$\boldsymbol{\nu}_N = \text{first N elements of } \boldsymbol{\nu}, \text{ remaining elements} = 0 \quad (33)$$

The number of multiplies in this process, particularly for large N_{taps}, is much less that $2N_{taps}^2$.

In the process given by equations (24)-(33), $\mathbf{b}$ is the vector $\mathbf{b}_0$ of the eight step algorithm described above, $\mathbf{y}$ is a vector of received symbols as described above, and $\mathbf{v}_N$ is the quantity $\hat{R}_{\tilde{y}\tilde{y}}[n]\mathbf{b}_0[n]$ in steps 2 and 7 of the conjugate gradient algorithm described above.

The conjugate gradient algorithm described above can be used to make other calculations. For example, the conjugate gradient algorithm described above can be used to calculate updated channel impulse response estimates. This calculation can be useful, for instance, because known alternative methods of calculating MMSE (Minimum Mean Square Error) equalizer tap weights are based on having a channel impulse response estimate.

The conjugate gradient method described above permits the continuous calculation of channel impulse response estimate updates between training sequences (when only a priori unknown symbols are received) in a receiver for a time varying multipath channel. This calculation utilizes a least squares channel impulse response estimation calculated from an over-determined system of equations. Dynamic changes to the channel impulse response may be tracked by using receiver trellis

decoder decisions on input symbols to form a long sequence of almost perfectly known symbols. This sequence should have relatively few errors.

The channel impulse response to be estimated is of length $L_h = L_{ha} + L_{hc} + 1$ where L_{ha} is the length of the anti-causal part of the channel impulse response and L_{hc} is the length of the causal part of the channel impulse response.

The reliable trellis decoder decisions on the input symbols is a vector designated herein as **s** of length L_s. A Toeplitz matrix S may be defined based on this vector **s** according to the following equation:

$$\mathbf{S} = \begin{bmatrix} S_{Lh-1} & S_{Lh-2} & - & - & - & - & - & S_0 \\ - & S_{Lh-1} & - & - & - & - & - & - \\ - & - & & & & & - & - \\ - & - & & & & & - & - \\ - & - & & & & & - & S_{Lh-1} \\ - & - & & & & & - & - \\ - & - & & & & & S_{Ls-Lh} & - \\ S_{Ls-1} & S_{Ls-2} & - & - & - & - & - & S_{Ls-Lh} \end{bmatrix} \qquad (34)$$

where the elements in the matrix of equation (34) are real and consist of the symbol decisions of vector **s**.

To ensure an over-determined system of equations, the following inequality is necessary:

$$L_S \geq 2L_h - 1. \qquad (35)$$

The matrix **S** is of dimension $(L_S - L_h + 1) \times L_h$ with $(L_S - L_h + 1) \geq L_h$. The received signal vector is **y** with data elements y_i for $L_{hc} \leq i \leq (L_S - L_{ha} - 1)$ where y_i is the received symbol corresponding to input symbol decision s_i. Then, $\mathbf{y} = \mathbf{Sh} + \mathbf{u}$ where **h** is the L_h long channel impulse response vector and **u** is a white noise vector.

The least squares solution for **h** is given by the following equation:

$$\hat{\mathbf{h}}_{LS} = \left(\mathbf{S}^T\mathbf{S}\right)^{-1}\mathbf{S}^T\mathbf{y}. \qquad (36)$$

By utilizing reliable trellis decoder input symbol decisions, there should be sufficient support for calculating a channel impulse response estimate of the required length. As required by inequality (35), the vector **s** of symbol decisions must be at least twice as long as the channel impulse response being estimated.

For an 8 VSB receiver application, the following parameters may be assumed: $L_h = 512$, $L_{ha} = 63$, $L_{hc} = 448$, and $L_S = 2496$. The vector **s** may be formed from a sequence of trellis decoder decisions on the input

symbols. Normally, the trellis decoder would just make output bit pair decisions, but it can also make equally reliable decisions on the input symbols.

The vector **s**, for example, may be selected as 3 segments (L_S = 2496 symbols) long. So, three data segments may be used to produce a single channel impulse response estimate update. A new channel impulse response estimate update can be obtained once per segment by proceeding in a sliding window manner. Optionally, several consecutive channel impulse response estimate updates can be averaged in order to further improve channel impulse response estimate accuracy if necessary. Of course, this additional step of averaging can be a problem if the channel impulse response is varying rapidly.

A vector **b** with fewer than 3 segments of symbol decisions may be used, but as stated in inequality (35), the length of vector **b** must be at least twice as long as the channel impulse response to be estimated. Longer **b** vectors help to diminish the adverse effect of additive white Gaussian noise on the channel.

The system of equations represented by equation (36) may be solved using the conjugate gradient algorithm in a manner similar to that previously described to solve

equation (11) for the MMSE tap weights where $\mathbf{S}^T\mathbf{y}$ in equation (36) takes the place of $\mathbf{r}_{s\tilde{y}}$ in equation (11), where $\mathbf{S}^T\mathbf{S}$ in equation (36) takes the place of $R_{\tilde{y}\tilde{y}}$ in equation (11), and where $\hat{\mathbf{h}}_{LS}$ in equation (36) takes the place of $\tilde{\mathbf{g}}$ in equation (11).

So, the conjugate gradient algorithm described above can facilitate the computation of $(\mathbf{S}^T\mathbf{S})\mathbf{b}$ for the case of the channel impulse response estimation just as it facilitated the computation of $\mathbf{R}_{\tilde{y}\tilde{y}}\mathbf{b}$ for the calculation of tap weights.

In further reducing the computation complexity necessary to implement the matrix-vector multiplications required to perform the conjugate gradient algorithm, such as when the conjugate gradient algorithm is used to estimate a channel as described above, it is useful to note that the conjugate gradient algorithm may be used to solve the following linear system for the channel estimate $\mathbf{h}$:

$$\mathbf{Sh} = \mathbf{y}. \qquad (37)$$

The matrix $\mathbf{S}$ is an m x n (e.g., 1985 x 512) Toeplitz matrix derived from symbol decisions, and $\mathbf{y}$ is a vector (the observation vector) derived from the received data.

The symbol decisions can be made, for example, by a trellis decoder or a slicer.

The system may be written as $\mathbf{S}^T\mathbf{S}\mathbf{h} = \mathbf{S}^T\mathbf{y}$ by multiplying both sides of equation (37) by $\mathbf{S}^T$. The conjugate gradient algorithm can be used to solve this system for the channel estimate $\mathbf{h}$ according to the following steps:

(1) Calculate the residual vector $\mathbf{r}_1$ representing the system $\mathbf{S}^T\mathbf{y} - \mathbf{S}^T\mathbf{S}\mathbf{h}_1$ as follows:

$$\mathbf{r}_1 = \mathbf{S}^T\mathbf{y} - \mathbf{S}^T\mathbf{S}\mathbf{h}_1 \qquad (38)$$

where $\mathbf{r}_1$ is a 512 x 1 vector, and where $\mathbf{h}_1$ is the previous channel estimate which may be initialized to any desired value such as $\mathbf{h}_1 = 0$.

(2) For k = 1 to n, iteratively calculate the following:

$$\text{(a)} \quad \mathbf{d}_k = \mathbf{r}_k + \beta_k \mathbf{d}_{k-1} \qquad (39)$$

where $\beta_1 = 0$, $\beta_{k\geq 2} = \dfrac{\mathbf{r}_k^T \bullet \mathbf{r}_k}{\mathbf{r}_{k-1}^T \bullet \mathbf{r}_{k-1}}$, and where $\mathbf{d}_k$ is a 512 x 1 vector;

$$\text{(b)} \quad \mathbf{h}_{k+1} = \mathbf{h}_k + \alpha_k \mathbf{d}_k \qquad (40)$$

where $\mathbf{h}_{k+1}$ is a 512 x 1 vector, where α_k is 1 x 1, where $\alpha_k = \frac{\mathbf{r}_k^T \bullet \mathbf{r}_k}{\mathbf{d}_k \bullet \mathbf{q}_k}$, where $\mathbf{q}_k = \mathbf{S}^T\mathbf{S}\mathbf{d}_k$, where $\mathbf{S}^T$ is a 512 x 1985 matrix, and where $\mathbf{S}$ is a 1985 x 512 matrix; and,

(c) $$\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k\mathbf{q}_k \qquad (41)$$

where $\mathbf{r}_{k+1}$ is a 512 x 1 vector.

Steps 2(a) - (c) are repeated enough times until a desired accuracy is achieved for the channel estimate h. The index k , for example, may be incremented up to a value of 5 although any other desired value can be used. It will be seen that step 2(b) includes the matrix vector multiplication $\mathbf{S}^T\mathbf{S}\mathbf{d}_k$. This operation is computationally very complex. A technique for reducing the computational complexity of implementing this operation is suggested above with respect to equations (23) - (33). The present invention improves this technique.

A straightforward approach to this technique would be to form the FFT of each of the three components $\mathbf{S}^T$, $\mathbf{S}$, and $\mathbf{d}_k$, point-wise vector multiply the three resulting FFTs, and then take the inverse FFT of the

multiplication result. This series of calculations, however, does not provide an accurate answer because the matrix **S** is not a true convolution matrix, and the IFFT of the triple product, therefore, produces a corrupted result. Figure 3 illustrates the matrix **S** which is derived from a vector of symbol decisions of length 2496 (three segments of a VSB field). The first row of the matrix S contains symbol decision 512 in the first column down to symbol decision 1 in the last column, the second row of the matrix S contains symbol decision 513 in the first column down to symbol decision 2 in the last column, and so on. The portions of the matrix **S** which prevent it from being a true convolution matrix are the triangular areas 40 and 42. The diagonal lines in Figure 3 are parallel.

According to the present invention, and as shown in Figure 4, matrix **S** of Figure 3 is decomposed to form two matrices $\hat{\mathbf{S}}$ and **F** such that the matrix $\hat{\mathbf{S}}$ is a true convolution matrix. Accordingly, the area 44 of the matrix $\hat{\mathbf{S}}$ is the same as the area 44 of the matrix **S**, and the symbol decisions in the areas 40 and 42 of the matrix $\hat{\mathbf{S}}$ are set to zero. The symbol decisions in the area 44 of the matrix **F** are set to zero, and the areas 40 and 42 of the matrix **F** are the same as the areas 40 and 42 of

the matrix **S**. A new linear system based on $\hat{\mathbf{S}}$ and **F** is then defined which may be solved for the channel estimate h. Advantageously, FFT operations of $\hat{\mathbf{S}}$ may be used for implementing the matrix vector multiplication of the conjugate gradient algorithm without corruption because $\hat{\mathbf{S}}$ is a true convolution matrix.

More specifically, because the linear system is defined by $\mathbf{y} = \mathbf{Sh}$, the following equations result:

$$\mathbf{y} = (\hat{\mathbf{S}} + \mathbf{F})\mathbf{h} \tag{42}$$

$$\mathbf{y} = \hat{\mathbf{S}}\mathbf{h} + \mathbf{F}\mathbf{h} \tag{43}$$

$$\mathbf{y} - \mathbf{F}\mathbf{h} = \hat{\mathbf{S}}\mathbf{h} \tag{44}$$

Defining $\mathbf{y} - \mathbf{F}\mathbf{h}_1 = \hat{\mathbf{y}}$, where $\mathbf{h}_1$ is a previous channel estimate derived by any known technique, the new system expression becomes $\hat{\mathbf{y}} = \hat{\mathbf{S}}\mathbf{h}$. This expression can be written as $\hat{\mathbf{S}}^T\hat{\mathbf{y}} = \hat{\mathbf{S}}^T\hat{\mathbf{S}}\mathbf{h}$. The conjugate gradient algorithm can then be applied to solve for **h** in the new system as follows:

(1) $\hat{\mathbf{y}} = \mathbf{y} - \mathbf{F}\mathbf{h}_1$,

$\mathbf{r}_1 = \hat{\mathbf{S}}^T\hat{\mathbf{y}} - \hat{\mathbf{S}}^T\hat{\mathbf{S}}\mathbf{h}_1$

(2) For k = 1 to n, iteratively calculate

$$(a) \quad \mathbf{d}_k = \mathbf{r}_k + \beta_k \mathbf{d}_{k-1}$$

$$(b) \quad \mathbf{h}_{k+1} = \mathbf{h}_k + \alpha_k \mathbf{d}_k$$

$$(c) \quad \mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k \mathbf{q}_{k-1}$$

where $\mathbf{h}_1$ may be initialized to any desired value such as 0, and where $\mathbf{d}_0$, $\mathbf{r}_o$, β_k, and α_k are as previously defined.

As indicated above following equation (40), step 2(b) includes calculating the expression $\mathbf{q}_k = \hat{\mathbf{S}}^T \hat{\mathbf{S}} \mathbf{d}_k$. This multiplication may be conveniently solved by forming the FFT of (i) the matrix $\hat{\mathbf{S}}$, (ii) $\mathbf{d}_k$, and (iii) $\hat{\mathbf{S}}^T$. The three FFT results are then point-wise vector multiplied, and the inverse FFT (IFFT) is taken of the multiplication result. The first N elements of this IFFT produces $\mathbf{q}_k$. N, for example, can have a value of 512 for a 1024 point FFT/IFFT. However, N can have other values as desired.

Figure 5 shows a system 50 for implementing this technique so as to determine a channel estimate h. The system 50 includes a decoder 52 that decodes the received data y to produce the symbols s. As described above, the decoder 52 may be a trellis decoder. However, the decoder 52 may be other forms of decoders such as slicers. A channel estimator 54 produces a channel estimate h using the techniques described above.

Modifications of the present invention will occur to those practicing in the art of the present invention. For example, the Fast Fourier Transform (FFT) and Inverse Fast Fourier Transform (IFFT) are used in the present invention as described above. Alternatively, the Discrete Fourier Transform (DFT) and Inverse Discrete Fourier Transform (IDFT) can be used instead of the FFT and IFFT.

Accordingly, the description of the present invention is to be construed as illustrative only and is for the purpose of teaching those skilled in the art the best mode of carrying out the invention. The details may be varied substantially without departing from the spirit of the invention, and the exclusive use of all modifications which are within the scope of the appended claims is reserved.